Electronic Banking

Aufgaben

Electronic Banking (EB) bezeichnet im Bankwesen die Abwicklung von Bankgeschäften über Internet mithilfe von elektronischen Endgeräten. Früher mussten Bankgeschäfte über Vordrucke (z. B. Überweisungsträger) abgewickelt werden, die den Banken als Buchungsbelege dienten. Durch Einführung elektronischer Zahlungssysteme wird der beleggebundene Zahlungsverkehr zunehmend bedeutungslos. Eine Privatbank will die Bankgeschäfte ihrer Privatkundinnen und -kunden deshalb auf EB umstellen.

1 Objektorientierte Entwicklung des E-Banking-Systems
Ein erstes Klassendiagramm finden Sie in Material 1.

1.1 Der Zugang einer Kundin oder eines Kunden (im Folgenden Kunde genannt) auf ein Konto[1] erfolgt über den EB-Server. Folgende Funktionalitäten sind bei der Entwicklung des EB-Systems bereit zu stellen:

- Der Kunde muss sich beim EB-Server unter Angabe von IBAN[2] und Online-PIN[3] des Kontos anmelden (Login).
- Er kann den aktuellen Kontostand abfragen.
- Der Kunde kann Überweisungen tätigen, wobei eine Überweisung nur ausgeführt wird, wenn das Konto des Kunden eine entsprechende Deckung aufweist.
- Der Kunde kann sich eine Übersicht über die Buchungen seines Kontos anzeigen lassen.
- Er kann nach Buchungen suchen.
- Der Kunde kann die Sitzung beenden.

Entwickeln und zeichnen Sie für diese Anwendungsfälle ein Anwendungsfalldiagramm in UML-Notation.

(5 BE)

1.2 Datenkapselung ist ein wichtiges Prinzip der Objektorientierten Programmierung. Beschreiben Sie den Zweck der Datenkapselung und die im Klassendiagramm (Material 1) vertretenen Zugriffsarten. Erläutern Sie das Geheimnisprinzip am Beispiel der Attribute `pin` und `kontostand` der Klasse `Konto`.

(4 BE)

1.3 In Material 2 ist ein Beispiel für die Buchungsanzeige eines Kontos dargestellt.
Zeichnen Sie ein UML-Objektdiagramm auf Grundlage dieser Buchungsanzeige unter Berücksichtigung des Klassendiagramms in Material 1.

Hinweis: Fehlende Attributwerte sind durch sinnvolle Werte zu ergänzen.

(4 BE)

---

[1] Um Überweisungen ausführen zu können, muss es sich bei dem Konto um ein Girokonto handeln. Zur Vereinfachung wurde im ersten Aufgabenteil allerdings auf eine Spezialisierung verzichtet.
[2] Die IBAN ist eine internationale standardisierte Notation für Bankkontonummern. Es handelt sich dabei um eine global eindeutige Bezeichnung, die auch die Bankleitzahl miteinschließt.
[3] Persönliche Identifikationsnummer (PIN) oder Geheimzahl zur Authentifizierung.

1.4 Implementieren Sie die Klassen `Konto` und `Buchung` aus Material 1 ohne get- und set-Methoden.

Hinweise: Der PIN eines neuen Konto-Objekts wird mithilfe der statischen Methode `generierePin()` der Klasse `EBVerwaltung` erzeugt. Die Methode `zeigeBuchungen()` der Klasse `Konto` generiert einen String in Form der Umsatzanzeige in Material 2. Die Methoden `einzahlen()` und `ueberweisen()` erzeugen nach erfolgreicher Ausführung der Operation jeweils ein Buchungsobjekt, das als `neueste` in der verketteten Liste eingefügt wird. Eine Überweisung wird nur ausgeführt, wenn das Konto dadurch nicht überzogen wird. Die Methode `sucheBuchungen()` durchsucht die Buchungstexte nach der übergebenen Zeichenfolge und gibt die gefundenen Buchungsobjekte als Liste zurück. Die `toString()`-Methode der Klasse `Buchung` generiert eine Zeichenkette mit den Buchungsdaten gemäß der Umsatzanzeige in Material 2. Die Dokumentationen der Klassen `Date`, `List` und `String` sind in Material 3 zu finden.

(14 BE)

1.5 Die Methode `sucheKonto()` der Klasse `EBVerwaltung` prüft zunächst die IBAN und sucht dann das entsprechende Konto. Dieses wird zurückgegeben, im Fehlerfall `null`.
Entwickeln Sie ein Struktogramm für den Algorithmus dieser Methode.

(5 BE)

1.6 Das EB-System ermöglicht einem Kunden den Zugriff auf sein Konto, um Überweisungen zu tätigen. Wurde vom EB-Server eine Verbindungsanfrage akzeptiert, werden Client-Anfragen entgegengenommen und gemäß Protokoll bearbeitet. Nach erfolgreicher Anmeldung kann der Kunde bei ausreichender Kontodeckung Überweisungen tätigen, bis er sich vom Server abmeldet. Eine typische Sitzung des EB sieht folgendermaßen aus:
(> für Antwort vom Server, < für Kommandos vom Client, <LF> Enter-Taste):

```
> +OK E-Banking
< anmelden;DE06550205000222222222;4554<LF>    (Client sendet IBAN und PIN)
> +OK Willkommen
< ueberweisen;DE68210501700012345678;Einkauf Topmarkt;99.76<LF>
  (IBAN des Empfängers, Buchungstext und Betrag)
> +OK Überweisung erfolgt
> Aktueller Kontostand: 108.04 EURO
< ueberweisen;DE68210501700012345678;KFZ-Versicherung;499.0<LF>
> -ERR Überweisung nicht erfolgt
> Aktueller Kontostand: 108.04 EURO
< quit<LF>
```

In der Beispielsitzung meldet sich ein Kunde mit der IBAN DE06 5502 0500 0222 2222 22 und dem PIN 4554 erfolgreich an und überweist 99,76 EURO auf das Konto mit der IBAN DE68 2105 0170 0012 3456 78. Für die zweite Überweisung besteht keine ausreichende Deckung auf dem Konto und es wird eine entsprechende Meldung ausgegeben. Daraufhin meldet sich der Kunde ab.

1.6.1 Erläutern Sie das Client-Server-Modell anhand des EB-Systems unter Berücksichtigung der Begriffe "Client", "Server", "Dienst" und "Protokoll".

(4 BE)

1.6.2 Implementieren Sie die Klasse `EBServer` aus Material 1 unter Berücksichtigung des in Aufgabe 1.6 wiedergegebenen Sitzungsprotokolls sowie des UML-Sequenzdiagramms für die Server-Seite aus Material 4.

Hinweise: Die Dokumentationen der Klassen `Socket`, `ServerSocket` und `String` sind in Material 3 zu finden. Zur Umwandlung eines Strings in eine Integer-Zahl bzw. Fließkommazahl stehen die Methoden `Integer.parseInt(s: String): int` und `Double.parseDouble(s:String): double` zur Verfügung.

(9 BE)

1.6.3 Entwickeln und zeichnen Sie das UML-Sequenzdiagramm für die Client-Seite unter Berücksichtigung des in Aufgabe 1.6 wiedergegebenen Sitzungsprotokolls sowie des UML-Sequenzdiagramms für die Server-Seite (Material 4) in Material 5.

Hinweise: Die Methode `getAnmeldeDaten()` der Klasse `EBClient` liefert einen String in der Form `anmelden;[iban];[pin]`, die Methode `getUeberweisungsDaten()` liefert einen String in der Form `ueberweisen;[text];[iban];[betrag]` bzw. `quit` für das Beenden der Sitzung.

(8 BE)

1.7 Der EB-Server soll zukünftig mehrere Clients gleichzeitig bedienen können.

1.7.1 Beschreiben Sie die Eigenschaften von Prozessen und Threads. Erläutern Sie anhand des E-Banking-Systems, dass es zu Problemen kommen kann, wenn mehrere Threads auf eine Variable zugreifen und benennen Sie eine Lösung für das Problem.

(4 BE)

1.7.2 Entwickeln Sie auf der Grundlage des Klassendiagramms aus Material 1 ein erweitertes Modell und zeichnen Sie dieses.

Hinweis: Es sind nur die für die Aufgabenstellung relevanten Klassen aus dem gegebenen Klassendiagramm (Material 1) zu übernehmen.

(3 BE)

2 Entwicklung einer Datenbank zur Verwaltung von Darlehensverträgen und Depotkonten
Ein vorläufiges Entity-Relationship-Modell (ER-Modell) ist in Material 6 dargestellt.

2.1 Überführen Sie das ER-Modell in das relationale Modell in der 3. Normalform.

Hinweis: Alle Relationen sind in der Schreibweise `Relation(PK, Attribut, ..., FK#)` anzugeben.

(4 BE)

2.2 Einige Daten der Datenbank sollen ergänzt, geändert bzw. ausgewertet werden. Folgende Funktionen stehen zur Verfügung:

- `NOW()` liefert das aktuelle Tagesdatum.
- `POWER(m,n)` liefert das Ergebnis von m^n.
- `ROUND(x,n)` liefert den auf *n* Nachkommastellen gerundeten Betrag *x*.

2.2.1 Die letzten zehn Buchungen des Kontos mit der IBAN DE06 5502 0500 0222 2222 22 sollen nach Buchungstag absteigend sortiert ausgegeben werden.
Geben Sie eine entsprechende SQL-Anweisung an.

(3 BE)

2.2.2 Der Kunde Paul Müller schließt heute ein neues Darlehen (Vertragsnummer 12345) über die Summe von 30.000 EUR ab. Die Laufzeit beträgt 10 Jahre bei einem Zinssatz von 1,76 Prozent. Die Raten werden im monatlichen Turnus (Turnus = 12) von dem Referenzkonto mit der IBAN DE06 5502 0500 0222 2222 22 abgebucht.
Entwickeln Sie die SQL-Anweisungen, um das neue Darlehen in die Datenbank einzufügen und die monatliche Rate zu setzen.

Hinweis: Die Mindest-Höhe *R* der (jährlichen) Annuität[4] eines Kredits mit der Kreditsumme S_0 bei einem Zinssatz von *i* (z.B. 3 Prozent ⇒ $i = 0{,}03$) und einer Laufzeit von *n* Jahren lässt sich mittels der Formeln

$R = S_0 \cdot \frac{(1+i)^n \cdot i}{(1+i)^n - 1}$ und $Rate = \frac{R}{turnus}$ berechnen.

(5 BE)

2.2.3 Am Monatsende werden unter anderem die Buchungen der monatlich zu zahlenden Darlehensraten durchgeführt. Dafür werden zuerst die Kunden mit der IBAN des Referenzkontos, der Vertragsnummer und der Ratenhöhe ausgegeben, bei denen aufgrund nicht ausreichender Deckung auf dem Konto keine Buchung erfolgen kann. Anschließend sollen die Kontostände aller Kunden, die ein Darlehen abgeschlossen haben, entsprechend aktualisiert werden.
Entwickeln Sie die entsprechenden SQL-Anweisungen.

Hinweis: Das Hinzufügen der Buchungen in die Buchungstabelle ist nicht Gegenstand dieser Aufgabe.

(6 BE)

2.2.4 Alle Kunden sollen mit der Gesamtsumme ihrer aufgenommenen Darlehen aufgeführt werden, sofern die Gesamtsumme 100.000 EUR übersteigt. Die Liste soll nach der Gesamtsumme absteigend sortiert sein.
Implementieren Sie eine entsprechende SQL-Anweisung.

(4 BE)

---

[4] Annuität ist eine regelmäßig jährlich fließende Zahlung, die sich aus Zins und Tilgung zusammensetzt. Tilgung bedeutet die Rückzahlung von Schulden.

2.3 Die Datenbank soll um folgende Anforderungen erweitert werden:

- Es sollen die beiden Darlehensarten Kleinkredit und Hypothekendarlehen unterschieden werden.
- Bei Kleinkrediten ist zusätzlich der Verwendungszweck zu speichern.
- Bei Hypothekendarlehen wird die Grundschuld und das Eigenkapital festgehalten. Außerdem wird jedes Hypothekendarlehen durch mindestens eine Immobilie mit den Eigenschaften Adresse, Immobilienart und Verkehrswert abgesichert. Eine Immobilie kann Sicherheit für mehrere Hypothekendarlehen sein.
- Ein Kunde kann Depotkonten eröffnen. Ein Depot zeichnet sich durch eine eindeutige ID und das Eröffnungsdatum aus. Auf einem Depotkonto werden die Bestände (Stückzahl) an Wertpapieren und alle Transaktionen verbucht. Ein Wertpapier hat eine eindeutige Wertpapierkennnummer (WKN), eine Bezeichnung und eine Währung. Die Transaktion zu einer Wertpapierposition eines Depots ist mit der Art (An- oder Verkauf), der Anzahl, dem Datum und dem Kurs zu speichern.
- Kaufpreise, Verkaufserlöse und Gutschriften eines Depotkontos werden über ein Girokonto des Kunden verbucht.

Entwickeln und zeichnen Sie ein ER-Modell, das die genannten Anforderungen umsetzt.

Hinweise: Die Vorlage des Entity-Relationship-Diagramms in Material 6 kann verwendet werden. Das ER-Diagramm ist mit Entitätstypen, Attributen und Beziehungen sowie deren Kardinalitäten in [min, max]-Notation darzustellen.

(13 BE)

2.4 Mitarbeiterinnen und Mitarbeiter der Bank beraten die Kunden in Finanzangelegenheiten. Dazu werden Beratungstermine vereinbart, die zurzeit in der folgenden Tabelle (Ausschnitt) festgehalten werden:

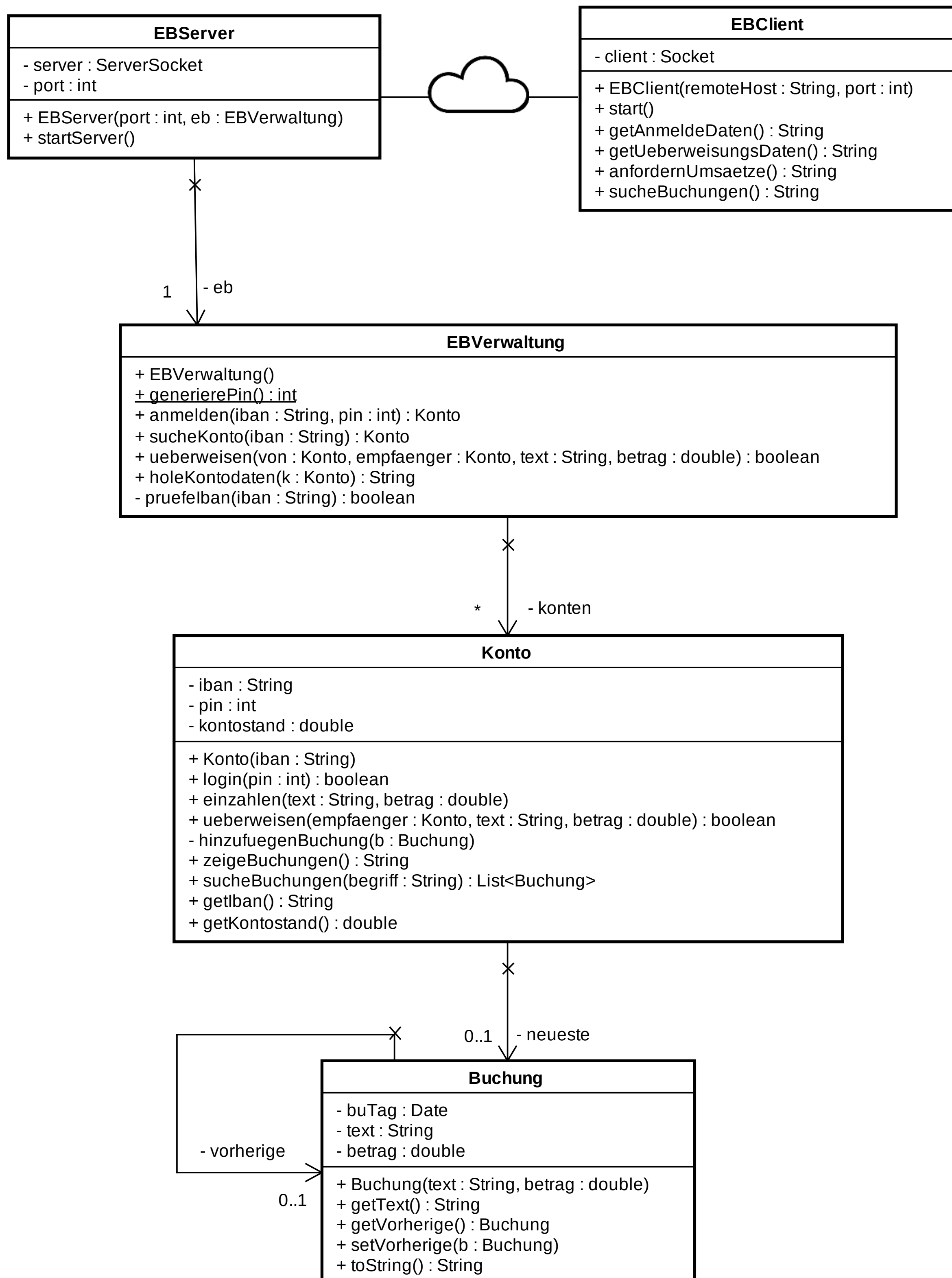
Berater	**Termin**	**Uhrzeit**	**Thema**	**Kunde**
Kai Schlau 0152 / 1234567	15.05.2022	16.30 – 17.30	Darlehensfinanzierung Wertpapierdepot	Hans Hauser Feldweg 5, 12345 Auheim 0173 / 66554489
	16.05.2022	19.00 – 21.00	Darlehensfinanzierung	Klara Kühn Lingelgasse 12, 23345 Lingen 0160 / 4566345
Klaus Meyer 0176 / 7199552	15.05.2022	10.00 – 12.00	Umschuldung Darlehensfinanzierung	Hans Hauser Feldweg 5, 12345 Auheim 0173 / 66554498
...	...	...	...	...

Analysieren Sie die Struktur der Tabelle und beschreiben Sie eventuell auftretende Probleme anhand von Beispielen.

(5 BE)

Material 1

UML-Klassendiagramm E-Banking (EB)



Material 2

Beispiel Buchungsanzeige

Konto IBAN DE07123412341234123412 Kontostand 599,48 EUR

Buchungstag	**Buchungstext**	**Betrag in EUR**
18.04.2022	Einkauf Proper-Markt	-42,70
10.04.2022	Geburtstagsgeld von Oma	100,00
01.04.2022	Mietzins 04/2022	-566,30
31.03.2022	Lohn 03/2022 Soft GmbH	1108,48

Material 3

Klassendokumentationen

Klasse `Date`

Date
+ Date() + Date(day : int, month : int, year : int) + addDays(numberOfDays : int) : Date + isBefore(d : Date) : boolean + isAfter(d : Date) : boolean + isBetween(d1 : Date, d2 : Date) : boolean + toString() : String

`Date()`
erzeugt ein `Date`-Objekt mit dem aktuellen Systemdatum.

`Date(day: int, month: int, year: int)`
erzeugt ein `Date`-Objekt mit den Werten von `day`, `month` und `year`.

`addDays(numberOfDays: int): Date`
liefert eine Kopie des `Date`-Objekts mit der addierten Anzahl von Tagen.

`isBefore(d: Date): boolean`
liefert `true`, wenn das `Date`-Objekt vor dem des Parameters `d` liegt

`isAfter(d: Date): boolean`
liefert `true`, wenn das `Date`-Objekt nach dem des Parameters `d` liegt

`isBeetween(d1: Date, d2: Date): boolean`
liefert `true`, wenn das `Date`-Objekt zwischen den Parametern `d1` und `d2` liegt.

`toString(): String`
liefert eine String-Repräsentanz des `Date`-Objekts im Format dd.mm.yyyy.

Klasse `List`

List<T>
+ List<T>() + add(obj : T) + add(index : int, obj : T) + contains(obj : T) : boolean + get(index : int) : T + remove(index : int) : T + remove(obj : T) : boolean + size() : int

`List<T>()`
erzeugt eine generische Liste mit Elementen des Typs `T`.

`add(obj: T)`
hängt das Objekt `obj` vom Typ `T` am Ende der Liste an.

`add(index: int, obj: T)`
fügt das Objekt `obj` vom Typ `T` an der Position `index` in die Liste ein.

`contains(obj: T): boolean`
liefert `true`, wenn das Objekt `obj` in der Liste enthalten ist, ansonsten `false`.

`get(index: int): T`
liefert das Listenelement an der Position `index` zurück bzw. `null`, falls `index` negativ oder größer gleich der Anzahl der momentan enthaltenen Elemente ist.

Material 3 (Fortsetzung)

`remove(index: int): T`
entfernt das Objekt vom Typ `T` an der Position `index` aus der Liste und gibt es zurück.

`remove(obj: T): boolean`
entfernt das Objekt `obj` aus der Liste. Falls `obj` mehrmals in der Liste enthalten ist, wird nur das erste Vorkommen entfernt. Der Rückgabewert ist `true`, falls das Objekt gefunden und entfernt wurde, sonst `false`.

`size(): int`
liefert die Anzahl der Elemente in der Liste zurück.

Klasse `ServerSocket`

ServerSocket
- localPort : int
+ ServerSocket(localPort : int) + accept() : Socket + close()

`ServerSocket(localPort: int)`
erzeugt einen Server-Socket und bindet ihn an den angegebenen Port.
`accept(): Socket`
wartet darauf, dass ein Client eine Verbindung aufbauen will.
Wurde eine Verbindung aufgebaut, liefert die Methode das entsprechende Socket-Objekt.
`close()`
schließt den Server-Socket.

Klasse `Socket`

Socket
- hostname : String - port : int
+ Socket(hostname : String, port : int) + connect() : boolean + readLine() : String + write(s : String) + close() : void

`Socket(hostname: String, port int)`
erzeugt ein Socket-Objekt.
`connect(): boolean`
stellt eine Verbindung zum Remote Host her; liefert `true`, wenn eine Verbindung hergestellt werden konnte, ansonsten `false`.
`readLine(): String`
liest eine Zeile vom Socket, bzw. liefert `null`, wenn der Socket nicht geöffnet ist. Eine Zeile wird durch ein Zeilenendezeichen abgeschlossen; das Zeilenendezeichen wird jedoch nicht in den zurückgegebenen String übernommen. Die Methode blockiert, bis eine komplette Zeile eingelesen ist.
`write(s: String)`
schreibt einen String zum Socket; ist die Schnittstelle nicht geöffnet geschieht nichts.
`close()`
löst die Verbindung auf.

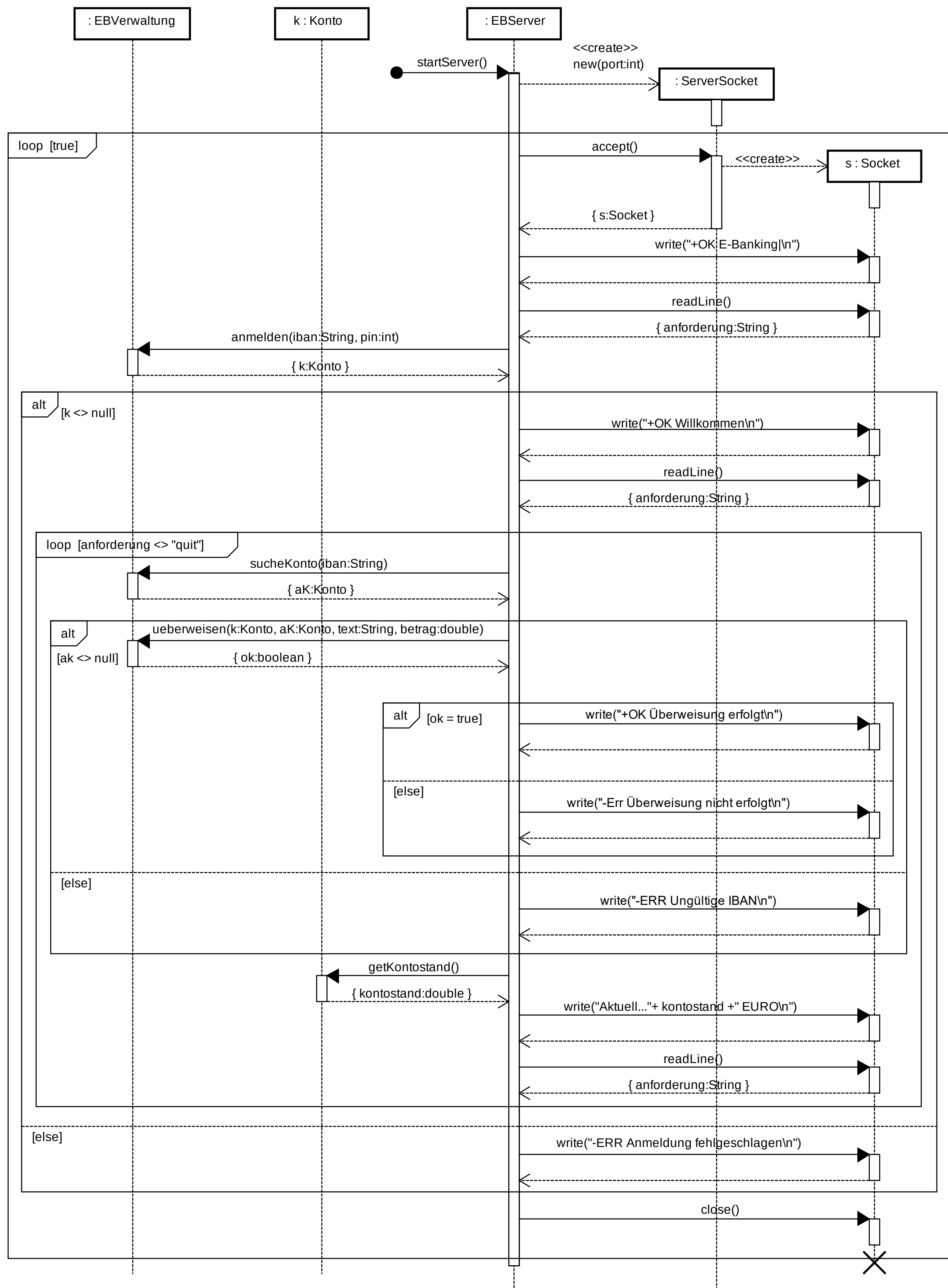
Klasse `String`

String
+ contains(sequence : String) : boolean + equals(str : String) : boolean + split(str : String) : String[] + startsWith(str : String) : boolean

`contains(sequence: String): boolean`
liefert `true`, wenn der String den übergebenen String `sequence` enthält, ansonsten `false`.
`equals(str: String): boolean`
liefert `true`, wenn beide Strings gleich sind, andernfalls `false`.
`split(str: String): String[]`
teilt einen String am Trennzeichen `str`. Die Teil-Strings werden in einem Feld zurückgeliefert.
`startsWith(str: String): boolean`
liefert `true`, wenn der String mit `str` beginnt, andernfalls `false`.

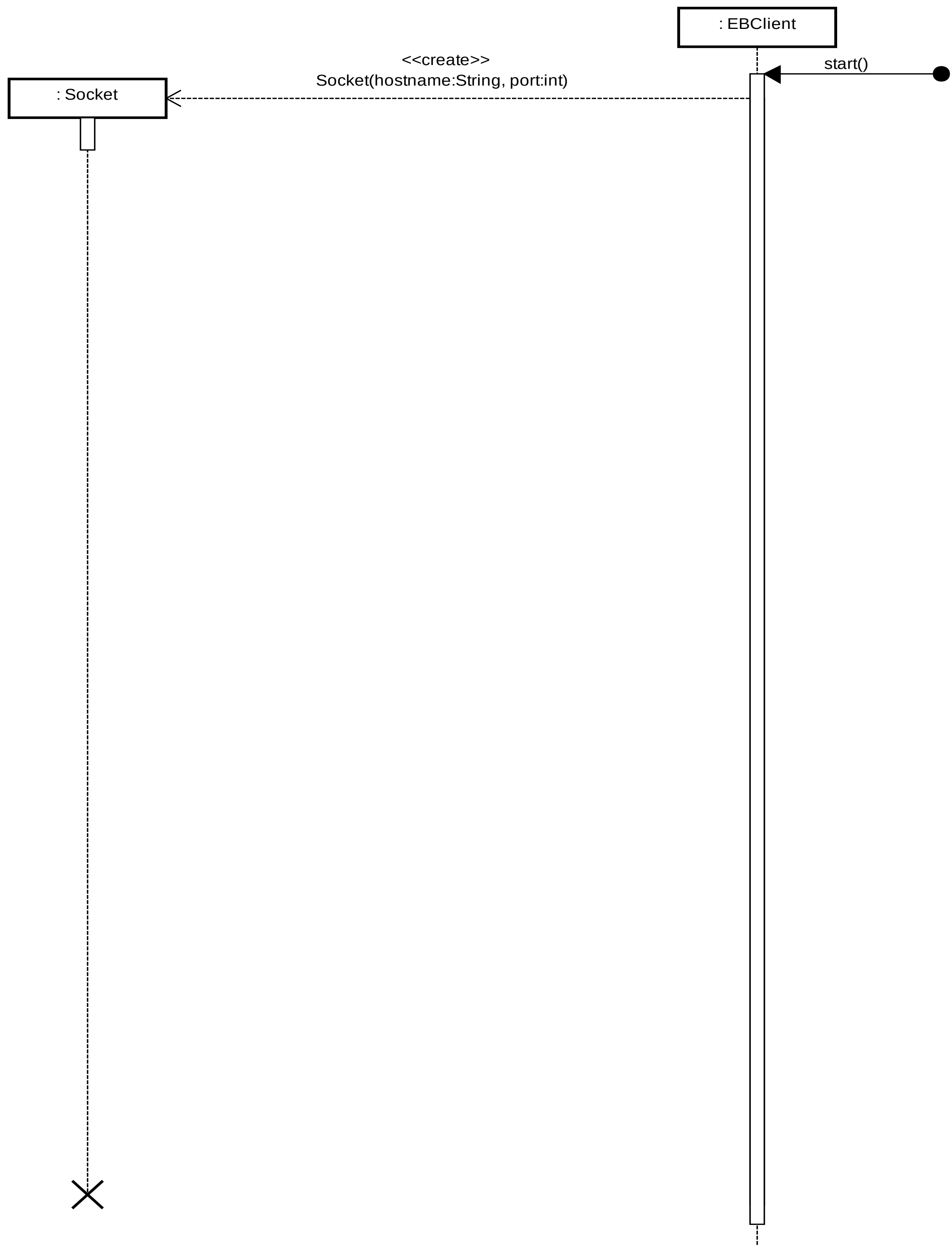
Material 4

UML-Sequenzdiagramm EB-Server



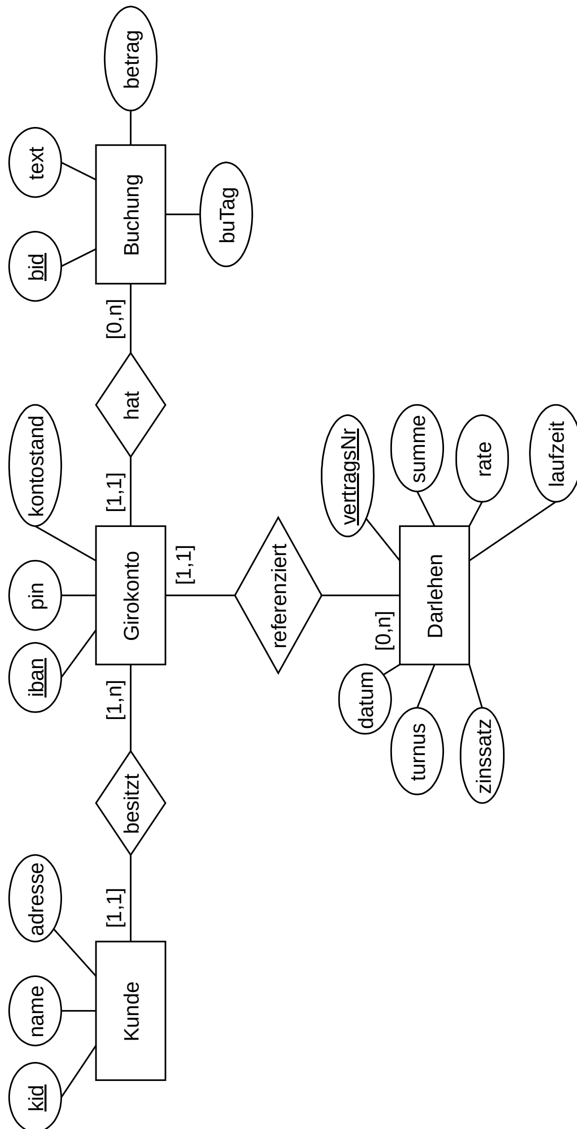
Material 5

Vorlage UML-Sequenzdiagramm EB-Client



Material 6

ER-Diagramm



Hinweise: Der Abschluss von Darlehensverträgen ist von der Führung eines Gehaltskontos (Girokonto) des Kreditnehmers bei der Bank abhängig. `Summe` legt die aufgenommene Kreditsumme fest. Die Darlehensrate (Attribut `rate`) setzt sich aus der Tilgungsrate und dem Zinsanteil zusammen. Mit Turnus wird die Anzahl der Ratenzahlungen pro Jahr bestimmt (12 monatlich, 4 vierteljährlich, 1 jährlich). Die Laufzeit wird in Jahren angegeben. Auf Wunsch des Kunden kann eine höhere als die Mindest-Rate vereinbart werden.